METHOD AND SYSTEM FOR INTELLIGENTLY FORWARDING MULTICAST PACKETS

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates generally to communication internetworking, and more specifically, to forwarding signals within a communications network.

Related Art

[0002] With the advent of the World Wide Web (WWW), global computer networks have quickly become cost-effective and reliable mediums for the exchange and management of information within an extensive array of computers and smaller computer networks. The computer networks vary in size and type such as, local internets, corporate intranets, local area networks (LAN), wide area networks (WAN), private enterprise networks, and the like. The global Internet is the most commonly known global computer network.

[0003] The evolution of global computer networks and supporting technologies have made it possible for government officials, educational institutions, businesses, nonprofit organizations, and individuals to communicate with the local networks or personal computers of other persons or organizations. The recreational and entertainment industries are also using global computer networks to expand their potential customer bases. As a result, more individuals and companies are using the Internet, for example, to transmit and/or multicast content for a variety of personal and business reasons. For example, a recording company may broadcast a live concert over the Internet to subscribers. As another example, a television production company may multicast a televised show over the Internet to a group of subscribers.

[0004] As more individuals and/or organizations take advantage of global computer networks to multicast content to a group of subscribers, greater emphasis must be placed on designing a distribution network capable of handling periods of heavy traffic. In a conventional multicast internet exchange, a network of routers are provided to transport the multicast content from a host-server to the client members of a group. Typically, the multicast packets are transmitted to every available router within a virtual local area network (VLAN). Upon receipt of the multicast packets, the routers must determine the destination and forward the packets downstream to the next router or client end station.

[0005] Flooding multicast traffic to every port within a VLAN is not the most efficient or cost-effective way to forward multicasts. As such, some multicast protocols have been developed to limit the multicast traffic to a few select ports within a VLAN. The ports are selected by determining whether the port communicates directly or indirectly with a client group member. An example of such a limited multicast protocol is described in Experimental Internet Protocol Standard, Request for Comments (RFC) 2362 (Internet Architecture Board) as Protocol Independent Multicast (PIM) Sparse Mode (SM).

[0006] However, even with a limited multicast protocol, there exists no conventional method for quickly, efficiently, and inexpensively forwarding a multicast packet by a layer 2 switch. The term "layer" is used herein to refer to, for example, a layer of the Open Systems Interconnection (OSI) seven-layer Reference Model. As apparent to one skilled in the relevant art(s), a layer 2 switch operates at the data link layer or layer 2 as defined by the OSI Reference Model. At layer 3 or network layer, as defined by the OSI Reference Model, forwarding activity is performed by a router.

[0007] Commercially available switches, such as those available from Cisco Systems, Inc., create a special label or tag for multicast packets. Although tag switching enables a router to read the tag and forward the packet, the router must be specially configured to be able to interpret the tag. A hub, a commonly used layer 2 device, broadcasts traffic to all interfaces. This is

expensive in terms of bandwidth usage, and inefficient as traffic is sent unnecessarily to interfaces or networks that may not require the traffic, thereby wasting network and CPU resources.

[0008] Other conventional network devices, such as routers, use the network layer to route and forward multicasts. Layer 3 processing requires additional processing time and memory. As such, like tag switching, the routing device must be specially configured to implement these processing requirements.

[0009] Therefore, a method and system are needed to address the above problems, and provide layer 2 processing to forward multicast packets efficiently and quickly.

SUMMARY OF THE INVENTION

[0010] The present invention solves the above problems by providing a method and system for intelligently forwarding multicast packets by a layer 2 switch. A switch is provided to implement layer 2 switching among a plurality of input/output ports that are connected to neighboring devices. The neighboring devices include other routers, end stations, and/or like network devices.

[0011] A packet processor is connected to the input/output ports and receives all content packets. The packet processor extracts a lookup key that is based on the destination address of the content packet. The switch of the present invention supports content packets from shared and explicit sources. As such, for shared source distributions, the packet processor derives a destination MAC address as the lookup key. For explicit source distributions, the packet processor derives a lookup key that is based on a source address, a destination address, a protocol type, and an incoming port, all associated with the content packet.

[0012] The lookup key is used to query a forwarding memory that contains a source-group table. The source-group table includes a listing of all multicast

groups that are being serviced by the switch. For shared source distributions, the forwarding memory also includes a forwarding table. Each entry in the forwarding table records a destination MAC address for a group and a corresponding outgoing port index. Similarly, for explicit source distributions, the forwarding memory also includes a session table. Each entry in the session table records, for each group, a source address, a destination address, a protocol type, an incoming port, and a corresponding outgoing port index.

[0013] Accordingly, packet processor queries forwarding memory which returns an outgoing port index. The outgoing port index serves as a lookup key to an outgoing port lookup table. The result of a lookup in the outgoing port lookup table is a list of outgoing port(s) that currently services the neighboring device(s) designated for the content packet.

[0014] The switch also receives and processes control messages that are used to configure (including, create, maintain and update) the forwarding memory and outgoing port lookup table. For instance, neighboring routers periodically exchange join/prune messages. Upon receipt, the switch of the present invention processes the join/prune messages to update the group lists stored or referenced in the forwarding memory and outgoing port lookup table. Neighboring routers also periodically exchange "hello" messages to announce their presence to each other. The switch processes the hello messages to build a neighbor list. The neighbor list specifies a source address and the incoming port that is connected to the source router of the hello message. This information is used to determine an outgoing port for a subsequent packet destined for that particular router.

BRIEF DESCRIPTION OF THE DRAWINGS/FIGURES

[0015] The accompanying drawings, which are incorporated herein and form part of the specification, illustrate the present invention and, together with the description, further serve to explain the principles of the invention and to

enable a person skilled in the pertinent art(s) to make and use the invention. In the drawings, like reference numbers indicate identical or functionally similar elements. Additionally, the leftmost digit(s) of a reference number identifies the drawing in which the reference number first appears.

[0016] FIG. 1 illustrates a layer 2 switch according to an embodiment of the present invention.

[0017] FIG. 2 illustrates an operational flow diagram for processing control packets according to an embodiment of the present invention.

[0018] FIG. 3 illustrates an operational flow diagram for intelligently forwarding a multicast packet according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

I. Introduction

[0019] The method and system of the present invention provides protocol snooping to intelligently forward multicast packets to those ports of a switch that are connected to a neighboring device that directly or indirectly services the members of a multicast group. In an embodiment, the protocol type is Protocol Independent Multicast (PIM). However the present invention supports other multicast protocols and/or standards.

[0020] Protocol snooping limits the multicast content to only those routers, within a VLAN domain, that require the content. Therefore, multicast traffic is not required to be seen by all multicast routers in the VLAN. As a result, the present invention can be implemented to conserve bandwidth on the VLAN because not all receivers are required to see the multicast traffic. The switch of the present invention also is not required to be configured for any special tagging technique because intelligent forwarding is based on information learned from control messages exchanged between the multicast routers.

II. System Overview

[0021] FIG. 1 illustrates a switch 100 according to an embodiment of the present invention. Switch 100 includes a plurality of input/output (I/O) physical ports 102a-102n, a plurality of packet processors 104a-104n, a forwarding content addressable memory (CAM) 106, an outgoing port lookup table (LUT) 108, a discovery list 110, a central processor (CPU) 112, a switch fabric 114, a shared memory buffer 116, a forwarding engine 118, and one or more neighboring switch interfaces (I/F) 120. The I/O physical ports 102a-102n (collectively referred to herein as I/O ports 102) serve as a physical layer interface supporting bi-directional communication between switch 100 and another neighboring device, such as an end station, a router, and/or like network device. Each individual I/O port 102 is connected to one or more neighboring devices. The connection between each I/O port 102 and the neighboring device(s) comprises wired, wireless, or both transmission media, including satellite, terrestrial (e.g., fiber optic, copper, coaxial, hybrid fiber-coaxial (HFC), or the like), radio, microwave, and/or any other form or method of transmission.

[0022] The I/O ports 102 are connected to packet processors 104a-104n (collectively referred to herein as packet processors 104). Each I/O port 102 exchanges packetized signals (e.g., electronic, electromagnetic, optical, or the like), demodulates the packetized signals, and delivers the packets to a packet processor 104. Each packet processor 104 parses and examines the packets for further processing. Further processing depends on whether the packets contain a control message and/or content payload(s).

[0023] If a packet contains content payload(s) (e.g., voice, data, and/or other forms of media, multimedia, or the like), packet processor 104 examines the packet to determine forwarding information (FID) regarding the disposition of the packet. The FID includes a destination port, port mirror requirement, packet type, VLAN handling, prioritization, multicast group membership,

and/or like features. The destination port indicates which of the plurality of I/O ports 102 will receive the packet. As described in greater detail below, packet processor 104 interacts with forwarding CAM 106, outgoing port LUT 108, and/or CPU 112 to determine the FID. Packet processor 104 postpends the FID to the packet before the packet is forwarded to switch fabric 114.

[0024] Forwarding engine 118 communicates with each packet processor 104, switch fabric 114, CPU 112, and other components of switch 100, as required. Forwarding engine 118 uses the FID, or the like, to direct traffic from one location to the next.

[0025] Switch fabric 114 is connected to shared memory buffer 116 and neighboring switch interface(s) 120. Switch fabric 114 stores packets in shared memory buffer 116. Shared memory buffer 116 assigns a shared memory location identifier (SMID) to each received packet, and stores the SMID in a priority queue located in the shared memory buffer 116. The priority queue is associated with the I/O port(s) 102 and/or neighboring switch interface(s) 120 designated in the FID. When the SMID reaches the front of the queue, the priority queue is emptied on a FIFO basis and the associated packet is forwarded to the associated I/O port(s) 102 and/or neighboring switch interface(s) 120. I/O port(s) 102 and/or neighboring switch interface(s) 120 operate to modulate packets and send packetized signals to a designated device(s).

[0026] In regards to I/O port(s) 102, the designated device(s) can be a neighboring end station, a router, and/or like network device, as described above. In regards to neighboring switch interface(s) 120, the designated device(s) is one or more neighboring switches, such as another switch 100, or like switching device(s). As described with respect to I/O ports 102, the connection between neighboring switch interface(s) 120 and a neighboring switch(es) comprises wired, wireless, or both transmission media, including satellite, terrestrial (e.g., fiber optic, copper, coaxial, hybrid fiber-coaxial (HFC), or the like), radio, microwave, and/or any other form or method of transmission.

[0027] On the other hand, if a packet received at packet processor 104 is a control message or if a content packet contains a control message (for example, in a header frame), packet processor 104 forwards the control message to discovery list 110 or CPU 112, as described in greater detail below. The present invention supports two types of control messages for determining FID: a hello message and a join/prune message. Both control message types are defined in Experimental Internet Protocol Standard, Request for Comments (RFC) 2362 (Internet Architecture Board). However the present invention includes any similar control messages created and/or exchanged to provide the functions described herein.

[0028] Accordingly, a neighboring router periodically transmits a hello message to switch 100 to announce its presence to switch 100 and/or other upstream designated router(s). Packet processor 104 forwards the hello message to discovery list 110. Discovery list 110 stores the hello message until it is retrieved by CPU 112, as described in greater detail below.

[0029] The second type of control message is a join/prune message. A neighboring router periodically exchanges join/prune messages with switch 100 and/or other neighboring router(s) to designate group memberships for a multicast. A join/prune message contains both a join set and a prune set. The join set lists the groups that receivers have requested, and the prune set lists the groups that are not required by receivers. Packet processor 104 forwards the join/prune message to CPU 112 for further processing, as described in greater detail below.

[0030] FIG. 1 is a conceptual illustration of switch 100 that facilitates explanation of the present invention. It would be apparent to one skilled in the relevant art(s) that one or more of the blocks can be performed by the same piece of hardware, module of software, or a combination thereof. It should also be understood that embodiments of the present invention can be implemented in hardware, software, or a combination thereof. In such an embodiment, the various components and steps would be implemented in hardware and/or software to perform the functions of the present invention. It

should be understood that either of discovery list 110, forwarding CAM 106, and outgoing port LUT 108 can be any type of memory, including RAM, SDRAM, or the like.

III. Constructing and Updating Forwarding Information

[0031] As described in reference to FIG. 1, packet processor 104 examines each content packet to determine its FID. The FID is postpended to the content packet, and switch 100 subsequently uses the FID to handle the disposition of the content packet. To determine the FID, packet processor 104 interacts with other system components to query one or more FID tables stored in the system memories (i.e., forwarding CAM 106, outgoing LUT 108, and/or discovery list 110). The FID tables are periodically populated and refreshed by control messages exchanged by the neighboring routers.

[0032] Referring to FIG. 2, flowchart 200 represents the general operational flow of an embodiment of the present invention. More specifically, flowchart 200 shows an example of a control flow for processing control messages to create or update forwarding information for multicasts.

[0033] The control flow of flowchart 200 begins at step 201 and passes immediately to step 203. At step 203, an I/O port 102 receives a packetized signal having a control message from a neighboring device. The physical form of the signal can be electronic, electromagnetic, optical, or the like. The signal is demodulated and delivered to packet processor 104.

[0034] At step 206, packet processor 104 determines the type of control message. As described above, in an embodiment, the control message is either a hello message or a join/prune message. If a hello message is detected, forwarding engine 118 sends the hello message to CPU 112, or stores the control packet in discovery list 110. Afterwards, the control passes to step 209.

[0035] At step 209, CPU 112 processes the hello message to create or update a neighbor list. The neighbor list identifies the neighboring router that sent the hello message, the address (e.g., IP address) of the neighboring router, the

incoming I/O port 102 that received the hello message, and the like. As such, switch 100 is able to track and update a list of IP addresses and I/O port(s) 102 that are associated with a neighboring router that transmits a hello message.

[0036] If the hello message is received from neighboring switch interface(s) 120, CPU 112 makes note that the corresponding router(s) is being serviced by a neighboring switch. Therefore, the incoming neighboring switch interface(s) 120 is noted in the neighbor list instead of noting an incoming I/O port 102.

[0037] On the contrary, if at step 206, a join/prune message is detected, the control passes to step 212. At step 212, the multicast mode of operation is considered. In an embodiment, switch 100 supports two operational modes for multicasting: shared source distribution multicasting and single source distribution multicasting. In an embodiment, shared source distribution multicasting is defined by protocol independent multicasting (PIM) sparse mode (SM) operational mode, and explicit source distribution multicasting is defined by PIM single source multicasting (SSM) operational mode. In an embodiment, the operational mode is established manually by a systems operator. In another embodiment, the operational mode is set and/or altered by CPU 112 and/or another application software in communication with CPU 112.

[0038] It should be understood, however, that the present invention supports other multicast protocols in addition to PIM SM and PIM SSM, as would be apparent to one skilled in the relevant art(s). For example, the switch of the present invention can be configured to support the Multiprotocol Label Switching (MPLS) standards defined by the Internet Engineering Task Force, the Point-to-Point Protocol (PPP) defined in Internet Protocol Standard (STD) 51, Request for Comments (RFC) 1661 (Internet Architecture Board), and/or like standards and/or protocols governing layer 2 switching.

[0039] Therefore, referring back to FIG. 2, the control passes to step 215 if the control message establishes shared source distribution trees (e.g., PIM SM operational mode), and the control passes to step 218 if the control message

establishes explicit source distribution trees (e.g., PIM SSM operational mode).

[0040] At step 215, forwarding engine 118 transfers the join/prune message from packet processor 104 to CPU 112. In turn, CPU 112 processes the join/prune message to construct or update one or more FID tables within forwarding CAM 106. Generally, the FID tables include a source-group table and a forwarding table and/or a session table. With respect to step 215, the FID tables include a source-group table and a forwarding table. The source-group table includes various data about the members of a multicast group. Such data includes, but is not limited to, a source address, destination address, and an outgoing port index associated with the source and/or destination address. The source address can be undefined or a wildcard since a multicast packet may have multiple sources. Accordingly, CPU 112 processes the join/prune message to update the aforementioned entries in the source-group table. CPU 112 also notes the incoming I/O port 102 that received the join/prune message and add an outgoing port index to the source-group table that identifies the incoming I/O port 102. Thereafter, CPU 112 also creates an entry in outgoing port LUT 108 to associate the outgoing port index to the receiving incoming I/O port 102.

[0041] Another FID table residing in forwarding CAM 106 is a forwarding table. Forwarding table comprises one or more forwarding entries that include a destination MAC address and an outgoing port index. Accordingly, CPU 112 updates the forwarding entries if the join/prune message is related to an existing destination MAC address.

[0042] Specifically, CPU 112 extracts multicast group information from the join/prune message to derive the destination MAC address. As described, the join list identifies the network addresses for neighboring routers that have been added to a distribution tree to support multicasts from the request group. The prune list, conversely, identifies the network addresses for the neighboring routers that will be removed from a distribution tree. Thus, the

prune list specifies a holdover period stipulating a time for discontinuing the membership of a neighboring router from a distribution tree.

[0043] Thus the multicast group information, extracted by CPU 112, includes network addresses (e.g., destination IP addresses) for designated neighboring routers. CPU 112 derives a destination MAC address from this multicast group information. In an embodiment, the MAC address is derived by reading the first three octets (bytes) of the multicast group address (e.g., IP address). As known to one skilled in the relevant art(s), the first three bytes of any multicast address are 01:00:5e. These three bytes are used as the first three octets in the MAC address. Next, the remaining three octets are constructed from the multicast group address which has 32 bits. Only the lower 23 bits of the multicast group address is used to construct the multicast MAC address.

[0044] Once constructed, the destination MAC address serves as a shared source lookup key. Accordingly, when packet processor 104 receives a multicast content packet, the destination MAC address is used as the lookup key to determine an outgoing port index for the received packet. The outgoing port index serves as a lookup key into outgoing port LUT 108. As such, outgoing port LUT 108 specifies one or more "outgoing" I/O ports 102 (and/or neighboring switch interface(s) 120) for each outgoing port index. When queried, outgoing port LUT 108 returns the outgoing I/O port(s) 102 (and/or neighboring switch interface(s) 120) to packet processor 104 to identify the destination neighboring device(s). Packet processor 104 postpends this forwarding information to the packet and forwards the packet to switch fabric 114, as described in greater detail below.

[0045] Therefore, after deriving the destination MAC address for the group, CPU 112 searches for a forwarding entry having a matching destination MAC address, and updates the forwarding table in forwarding CAM 106. This is accomplished by CPU 112 retrieving the associated outgoing port index. Thereafter, CPU 112 utilizes the outgoing port index to query the associated tuple in outgoing port LUT 108. CPU 112 then updates the tuple by revising the associated outgoing ports (i.e., I/O port(s) 102 and/or neighboring switch

interface(s) 120) according to the designated routers and/or other device(s) in the new group.

[0046] As the multicast group information comprises network addresses, CPU 112 performs layer 3 processing to derive the MAC address. CPU 112 thus populates forwarding CAM 106 with the layer 3 information. Therefore, when packet processor 104 queries forwarding CAM 106, packet processor 104, in essence, utilizes information derived from layer 3 processing (i.e., protocol type and network address) to intelligently forward a packet during layer 2 processing.

[0047] As described, the present invention supports both shared trees and explicit source trees. For shared trees, forwarding is based on the MAC destination address lookup described in reference to step 215. However, for a source specific tree, a session lookup methodology is implemented as described with respect to step 218.

[0048] At step 218, the join/prune message has specified an explicit source distribution tree (e.g., PIM SSM operational mode). Hence, forwarding engine 118 transfers the join/prune message from packet processor 104 to CPU 112. In turn, CPU 112 processes the join/prune message to construct or update the FID table(s). With respect to step 218, the FID tables include a source-group table and a session table. The source-group table includes various data about the members of a multicast group, as described above in reference to step 215. The session table includes a session entry comprising a multicast source IP address, a destination IP address, an incoming port (i.e., I/O port 102 or neighboring switch interface 120), and protocol type as part of the explicit source lookup key, and outgoing port index being the result of lookup. The incoming port is used to specify which I/O port(s) 102 (and/or neighboring switch interface(s) 120) are eligible for a session match once a packet arrives on an I/O port 102.

[0049] Therefore, when processing a join/prune message, CPU 112 derives the explicit source lookup key information from the control message, and searches for a matching session entry. If an matching entry is determined,

CPU 112 retrieves the associated outgoing port index. Thereafter, CPU 112 utilizes the outgoing port index to query the associated tuple in outgoing port LUT 108. CPU 112 then updates the tuple by revising the associated outgoing ports (i.e., I/O port(s) 102 and/or neighboring switch interface(s) 120) according to the designated routers and/or other device(s) in the new group.

[0050] As described in detail below, a session entry is queried, or created if necessary, when a multicast content packet is received by packet processor 104 and forwarded to CPU 112. The outgoing port index, resulting from a lookup in the session table, serves as a lookup key into outgoing port LUT 108. The list of outgoing ports (i.e., I/O port(s) 102 and/or neighboring switch interface(s) 120) are constructed and maintained according to join/prune messages as described above.

[0051] Once discovery list 110, forwarding CAM 106, and/or outgoing port LUT 108 have been properly constructed or updated, the control passes to step 221. At step 221, CPU 112, interacting with forwarding engine 118, either returns the hello message or join/prune message to packet processor 104, or forwards the control message to switch fabric 114. If the control message is destined for another router within the VLAN domain, the control message is associated with a SMID, queued according to the designated neighboring router, and forwarded to the router so that the neighboring router can update its forwarding table or neighbor list. After the packet has been transmitted to the designated neighboring router(s), the control flow ends as indicated by step 295.

IV. Intelligent Forwarding Shared and Explicit Source Multicasts

[0052] Referring to FIG. 3, flowchart 300 represents the general operational flow of an embodiment of the present invention. More specifically, flowchart 300 shows an example of a control flow for intelligently forwarding multicast content packets in shared source distribution multicast operational mode or explicit source distribution multicast operational mode.

[0053] The control flow of flowchart 300 begins at step 301 and passes immediately to step 303. At step 303, an I/O port 102 receives a packetized content signal from a neighboring device. The physical form of the signal can be electronic, electromagnetic, optical, or the like. The signal is demodulated and delivered to packet processor 104.

[0054] At step 306, packet processor 104 determines whether the signal comprises a unicast content packet or a multicast content packet. In an embodiment, packet processor 104 examines the destination address to determine if the packet is a unicast packet or a multicast packet. For instance, a multicast content packet has a Class D IP destination address ranging from "224.1.0.0" to "239.255.255.255." If packet processor 104 detects a unicast packet, the control flow passes immediately to step 321, as described below. Otherwise, the control flow passes to step 309 for multicast processing.

[0055] At step 309, packet processor 104 derives or reads a lookup key. As described in reference to FIG. 2, switch 100 supports two modes of operation: shared source distribution multicasting and explicit source distribution multicasting.

[0056] Referring back to step 309, if switch 100 is operating with shared source distributions (e.g., PIM SM), packet processor 104 extracts a shared source lookup key. Packet processor 104 reads or derives a destination MAC address from the content packet, and the destination MAC address is used as the shared source lookup key. In an embodiment, the destination MAC address is derived by reading the first thee octets and constructing the remaining three octets from the multicast group address, as described above in reference to step 215 in FIG. 2.

[0057] On the other hand, if, at step 309, switch 100 is operating with explicit source distributions (e.g., PIM SSM), packet processor 104 extracts an explicit source lookup key from the content packet. The explicit source lookup key is based on the source IP address, destination IP address, protocol type derived from the multicast packet, and the incoming I/O port 102 that received the packet.

[0058] At step 312, packet processor 104 utilizes the lookup key to query a FID table stored in forwarding CAM 106. More specifically, for shared source distributions, packet processor 104 utilizes the destination MAC address as the lookup key to query a forwarding table located in forwarding CAM 106. For explicit source distributions, packet processor 104 utilizes the explicit source lookup key to query a session table located in forwarding CAM 106. If a match is found, the control flow passes immediately to step 321, as described below. Otherwise, the control flow passes to step 315.

[0059] At step 315, the content packet is delivered to CPU 112 for further processing since no match exists for the extracted lookup key. Using other information extracted from the content packet, CPU 112 queries the source-group table residing in forwarding CAM 106. In an embodiment, CPU 112 uses the destination group address and, if available, the destination source address(es) to locate a matching entry. If a match is determined, CPU reads the corresponding outgoing port index, and control passes to step 318.

[0060] At step 318, CPU 112 creates a new entry in either the forwarding table or session table. For shared source distributions, a new forwarding entry is added to the forwarding table based on the derived destination MAC address. For explicit source distributions, a new session entry is added to the session table based on the session lookup key. From the information returned from the source-group table, CPU 112 adds the returned outgoing port index to the forwarding entry or session entry.

[0061] However, if a match is not determined at step 315, control passes to step 317. At step 317, CPU 112 identifies the outgoing I/O port(s) 102 (and/or neighboring switch interface(s) 120) that currently services the destination router(s) or other network device(s) associated with the destination address from the content packet. To identify the outgoing I/O port(s) 102 (and/or neighboring switch interface(s) 120), CPU 112 queries discovery list 110 to determine if a hello message has been received from the destination router(s). If a hello message has been received, CPU 112 reads the outgoing I/O port(s)

102 (and/or neighboring switch interface(s) 120) from the neighbor list. Thereafter, control passes to step 321.

[0062] If no hello message has been received from the destination router(s), the packet is dropped for that router(s), and the control flow passes immediately to step 395. However, in an alternative embodiment, the packet is multicast to all routers within the VLAN if no hello message has been received from the destination router(s). As such, CPU 112 would identify all outgoing I/O port(s) 102 (and/or neighboring switch interface(s) 120) that service each router within the VLAN. In this embodiment, the control flow would then pass to step 321.

[0063] At step 321, packet processor 104 receives the outgoing I/O port(s) 102 and/or neighboring switch interface(s) 120 for the content packet. More specifically, if a lookup match is found at step 312, forwarding CAM 106 returns the outgoing port index corresponding to the lookup key (i.e., destination MAC address or explicit source lookup key). Subsequently, outgoing port LUT 108 is queried to identify the outgoing I/O port(s) 102 and/or neighboring switch interface(s) 120 corresponding to the outgoing port index. The corresponding list of outgoing I/O port(s) 102 and/or neighboring switch interface(s)120 is sent to packet processor 104.

[0064] However, if a new table entry is created at step 318, the corresponding outgoing I/O port(s) 102 and/or neighboring switch interface(s) 120 are retrieved from outgoing port LUT 108 based on the outgoing port index returned from the source-group table. Additionally, if the corresponding outgoing I/O port(s) 102 and/or neighboring switch interface(s) 120 are determined from the neighbor list as described at step 317, the corresponding list is simply forwarded to packet processor 104.

[0065] If a unicast packet is detected at step 306, then, at step 321, forwarding CAM 106 returns the FID for the packet. The FID includes the outgoing I/O port(s) 102 or neighboring switch interface(s) 120 for the destination neighboring device. The FID is sent to packet processor 104, and the control flow passes to step 324.

[0066] At step 324, packet processor 104 receives the list of outgoing I/O port(s) 102 and/or neighboring switch interface(s) 120 corresponding to the content packet. Packet processor 104 postpends the list to the packet.

[0067] At step 327, forwarding engine 118 interacts with packet processor 104 to forward the packet to switch fabric 114. At step 330, switch fabric 114 creates a SMID for the packet, and stores the packet in shared memory buffer 116. Switch fabric 114 also stores the corresponding SMID in a priority queue also located in shared memory buffer 116. At step 333, switch fabric 114 sends the packet to the outgoing I/O port(s) 102 and/or neighboring switch interface(s)120 linked to the designated neighboring device(s). The outgoing I/O port(s) 102 and/or neighboring switch interface(s)120 modulate the packetized signal and transmit the signal to the designated neighboring device(s). After the packet has been transmitted to the designated neighboring device(s), the control flow ends as indicated by step 395.

V. Conclusion

[0068] FIGs. 1-3 are conceptual illustrations that facilitates explanation of the present invention. It will be apparent to one skilled in the relevant art(s) that the same piece of hardware or module of software can perform one or more of the blocks. It should also be understood that embodiments of the present invention could be implemented in hardware, software, or a combination thereof. In such an embodiment, the various components and steps would be implemented in hardware and/or software to perform the functions of the present invention.

[0069] While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. It will be apparent to persons skilled in the relevant art(s) that various changes in form and detail can be made therein without departing from the spirit and scope of the invention. Moreover, it should be understood that the method and system of the present invention

should not be limited to a network with one layer 2 switch. The present invention can be implemented in any network of routers interconnected via several layer 2 switches in the core. Thus, the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.